information

MDPI

Article

Word Sense Disambiguation Studio: A Flexible System for WSD Feature Extraction †

Gennady Agre [1,*], Daniel Petrov [2] and Simona Keskinova [2]

[1] Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, 1113 Sofia, Bulgaria
[2] Laboratory of Computer Graphics and Geographical Information Systems, Technical University of Sofia, 2173 Sofia, Bulgaria; desolatora93@gmail.com (D.P.); simona181212@gmail.com (S.K.)
* Correspondence: agre@iinf.bas.bg; Tel.: +359-2-870-01-18
† This paper is an extended version of our paper presented in 18th International Conference AIMSA 2018, Varna, Bulgaria, 12–14 September 2018.

Received: 27 January 2019; Accepted: 27 February 2019; Published: 5 March 2019

Abstract: The paper presents a flexible system for extracting features and creating training and test examples for solving the all-words sense disambiguation (WSD) task. The system allows integrating word and sense embeddings as part of an example description. The system possesses two unique features distinguishing it from all similar WSD systems—the ability to construct a special compressed representation for word embeddings and the ability to construct training and test sets of examples with different data granularity. The first feature allows generation of data sets with quite small dimensionality, which can be used for training highly accurate classifiers of different types. The second feature allows generating sets of examples that can be used for training classifiers specialized in disambiguating a concrete word, words belonging to the same part-of-speech (POS) category or all open class words. Intensive experimentation has shown that classifiers trained on examples created by the system outperform the standard baselines for measuring the behaviour of all-words WSD classifiers.

Keywords: word sense disambiguation; word embedding; classification; neural networks; random forest; deep forest; JRip

1. Introduction

The task of word sense disambiguation (WSD) is to computationally identify the correct meaning of a word by its use in a particular context [1]. The complexity of this task is due to such reasons as the lack of a unified representation for word senses, the use of different levels of granularity of sense inventories, a strong dependence of the task on available knowledge resources and so forth. WordNet [2], which is currently used as a standard word sense inventory for English texts, relates each word (word lemma) to a set of WordNet sense identifiers and its latest version (WordNet 3.1) contains about 155,000 words organized across 117,000 synsets.

Based on the knowledge resources used, the existing approaches for solving the WSD task can be split into two main groups—supervised and knowledge-based. The supervised approaches consider WSD as a classic machine learning task—they need sense annotated collections of texts (corpora) for learning a classification model, which then are applied to finding a proper sense of a target word in a previously unseen context. Since the manual annotation of texts is a very time consuming task, most of the existing supervised WSD systems for English use the largest manually annotated corpus—SemCor [3] as their training set, which contains around 234,000 words that are manually annotated with part-of-speech (POS) tags and word senses from the WordNet inventory.

A semi-supervised approach is a variant of the supervised approach to WSD, which tries to avoid the necessity of the manual creation of sense annotated collections of training texts. It does so by constructing artificial, automatically sense annotated corpora or by exploring such corpora in conjunction with manually annotated ones. It has been shown that in some settings such semi-supervised approaches allow creating training corpora that can be used for learning classification models outperforming similar models trained on manually annotated ones [4,5].

Knowledge-based approaches do not need a sense annotated corpus and rely only on lexical recourses such as a dictionary or a computational lexicon. The idea is to take advantage of the structural and lexical-semantic information in such resources to choose among the possible senses of a target word occurring in context. The Lesk algorithm [6] is one of the earliest approaches in this group. It is based on the computation of overlap between the context of the target word and its definitions from the sense inventory. An important branch of knowledge-based systems are graph-based systems that use some structural properties of semantic graphs for solving the WSD task [7,8]. Such systems first create a graph representation of the input text and then traverse the graph by different graph-based algorithms (e.g., PageRank) in order to determine the most probable senses in the context. The knowledge-based systems usually have lower accuracy than the supervised systems since they tend to adopt bag-of-word approaches which do not exploit the local lexical context of a target word, including functional and collocation words [5]. However, they have an advantage of a wider coverage, thanks to the use of large-scale knowledge resources [1].

The present paper is devoted to describing a system called *WSD Studio*, which is intended for the extraction of various features and creating sets of training and test examples for solving the WSD task based on the supervised approach. The main characteristics of the approach applied by the system have been already presented in Reference [9], so in this paper we make more emphasis on the functional architecture of the system, on newly conducted experiments and on deeper discussion of the results. The structure of the paper is as follows: the next section discusses some related work. Section 3 is devoted to the detailed description of the system's functional architecture. The experiments with different classification models learned from the sets of training examples constructed by the system are presented in Section 4. Section 5 is devoted to the discussion and presentation of our plans for further development of the system. The last section concludes the paper.

2. Related Work

There are two variants of the WSD task—the lexical sample WSD and the all-words WSD. In the first case only a restricted subset of target words should be disambiguated. Such words usually occur one per sentence. The all-words WSD task requires finding senses for all open class words in a sentence (i.e., nouns, verbs, adjectives and adverbs). The complexity of this task is higher because of:

1. *A huge number of classes*: for example, a WordNet-based dictionary used for sense annotating a text contains about 150,000 open class words with about 210 000 different senses (classes).
2. *Data sparseness*: it is practically impossible to have a training set of adequate size that covers all open class words in the dictionary, which (for the supervised approach to WSD) makes it impossible to create a good classification model for disambiguating all words.

To avoid these problems, the existing supervised WSD systems try to solve the all-words WSD task by constructing a set of classifiers (so called *word experts*)—one specialized classifier per open class word or even per each word category (noun, verb, adjective or adverb). A training example for such a word expert is represented by a set of features extracted from the context of the corresponding word (a fixed number of words occurring left and right to the target word in a sentence) and labelled by a concrete sense of the word (class) selected from a set of possible word senses presented in the dictionary.

One of the state-of-the-art supervised all-words WSD systems is IMS [10]. It constructs a classifier for each type of open class word w and applies a bag-of-words approach for representing examples.

Each example (an occurrence of w in the text) is represented by three sets of features extracted from the context of w (a window with the size of three words left and right to w). The first set consists of POS tags of the target and context words, belonging to the same sentence as the target word. The second set contains a special binary representation of surrounding words in their lemma forms and the third set consists of a binary representation of 11 features—local collocations between context words. The examples constructed by IMS can be used by different classifiers and the best results have been achieved by a linear kernel support vector machine (SVM) [11].

A present tendency in the development of supervised WSD systems is to use word embeddings instead of (or along with) "traditional" discrete representations of words. Word embeddings are a distributional representation of words that contains some semantic and syntactic information [12]. There are two approaches for integrating supervised WSD systems with word embeddings. The first approach proposes different variants of including word embeddings into binary representation of examples constructed by the IMS system [13–15]. In most cases different combinations of word embeddings of context words were used as additional features. The extensive experiments have shown [16] that such integration is able to improve the accuracy of IMS on the all-words WSD task for several benchmark domains.

All such IMS-based approaches for solving the all-words WSD task suffer from the following shortcomings:

- Necessity to construct separate classification models for each open class word (or even for each word POS category).
- Relying on rather complicated procedures for extracting binary features and their integration with word embeddings.
- Ignoring the order of words in the context.

An alternative approach for using word embeddings for the WSD task that can be seen as an attempt to overcome the last two shortcomings is the collocation approach, in which embeddings of words at specific positions near the target word are directly used as features of examples. In most cases classification models based on this approach are learned by specially designed recurrent neural networks [17]. The classification of an unseen target word is based on a simple idea—the input context of the word is transformed into a vector in the word embedding space, which is then mapped by the model to the space of possible word sense vectors represented by sense embeddings. Such systems usually use Bidirectional LSTM networks as a classifier [18–20] and have achieved promising results on the lexical sample task. However, such systems have to construct different models (with different sets of parameters) for each target word, need sense embeddings for representing examples and demand enormous computational power and time to be trained. A promising research in this direction is presented in Reference [21], where an echo state network architecture is proposed for solving the WSD task.

3. WSD Studio—Architecture and Functionality

The WSD Studio is a flexible system for extracting features and constructing examples to be used for solving the all-words WSD task in the supervised manner. The examples are created according to the collocation approach and can be used by classifiers of all kinds. The system requires the following knowledge resources as input:

- *A WordNet-based dictionary*—a file that relates open class words (in their lemma forms) to their possible senses. The dictionary is expected to be a text file, where each line represents a word and its senses in the following format: *{word}* $\{sense_1\}$:$\{tag\text{-}count_1\}$ $\{sense_2\}$:$\{tag\text{-}count_2\}$ ... $\{sense_n\}$:$\{tag\text{-}count_n\}$
- where *{word}* is a WordNet word, *{sense}* is the *synset_id* of the sense in the WordNet and *{tag-count}* is the tag count of the sense in the WordNet. Currently the WSD Studio supports a variation of the *synset_id*, where the first digit of the *synset_id* that represents the POS category is removed

and instead a dash and the Word-Net indicator *ss_type* are appended at the end. An example of *synset_id* transformation looks like this: 105,820,620 -> 05820620−n.

- *Training and test text data* that may consist of one or several text files containing a set of sentences (If the training text data is not available, the system uses SemCor [3] as the default training set). Each sentence is a sequence of words in their *lemma forms* that may be annotated by their POS categories. Each open class word in a sentence should be annotated by a sense from the corresponding WordNet-based dictionary. The system accepts two formats for representing training and test text data: the Unified Evaluation Framework (UEF) XML format [16] and a plain text alternative. Compared to the UEF XML, in the plain text format each annotated text is expected to be in a different text file. Each word occupies one line, where each open class word is followed by an interval and the *synset_id* of its sense. Each sentence ends with an empty line. When datasets in the UEF XML format are used, an additional text file containing mappings from *sense_index* to *synset_id* must be provided. Each line must contain one mapping pair with the *sense_index* at the beginning of the line followed by an interval and the *synset_id*.
- *Word embeddings*—a file relating words with their embeddings.
- *Word sense embeddings (optional)*—a file relating word sense identifiers with their embeddings.

The functional architecture of the system is shown on Figure 1. At the beginning, a WordNet-based dictionary along with training and test text data are loaded to the Data Analyzer module, which parses, analyses and saves the input data into a special intermediate format. The module calculates some basic statistics (at the level of sentences and at the level of words) providing the user with information which can facilitate the choice of some parameters of the representation of training and test examples to be created by the system. The Data Generator module is the central module of the system that allows the user to extract in a flexible way various features from the training text data, depending on the parameters set by the user, as well as to generate training and test examples combining these features in a preferred by the user order and format. Finally, the constructed examples are transferred to the Data Writer module, which saves them as a set of training and test files in a format preferred by the user and accompanied by various baseline accuracy statistics to be used for WSD classifier evaluation. Such files can then be used to learn (outside the system) classification models for solving the all-words WSD task.

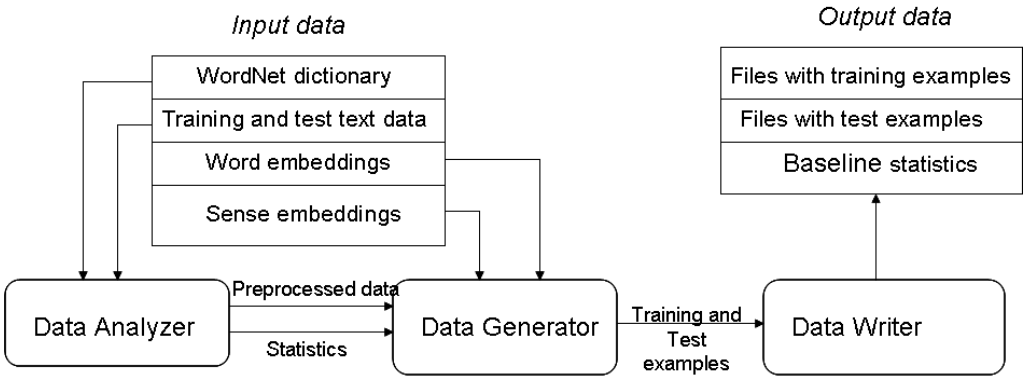

Figure 1. The functional architecture of the WSD Studio.

3.1. Data Analyzer Module

The DataAnalyser module parses, analyses and saves the input data into a bundle called "project," written in an intermediate format. Such a solution has the following advantages:

- The process of data generation becomes format independent and the specifics of input format parsing are encapsulated only in the project creation algorithm.
- Any potential problems related to data parsing are avoided since all data used by the data generation algorithm is saved in an easy processable/parsable way.

- The bundling of input data into a data project allows the user to easily create several data configurations and to reuse them multiple times.

The analysis consists of a dictionary analysis and a dataset analysis. The dictionary analysis calculates some basic statistics such as the number of words in the dictionary, the number of words per given sense, the maximum number of senses per word and so forth. One of the results of this analysis is a list of monosemous words (i.e., words with only one meaning) for which no examples will be constructed by the system since their meaning does not depend on any context and can be determined directly from the dictionary.

During the dataset analysis the module also calculates a number of occurrences of polysemous words (i.e., words with several meanings) in the training and test data (which corresponds to the numbers of training and test examples), frequencies of different word senses for each polysemous word in the training data and so forth. The module also calculates a number of occurrences of two special types of polysemous words—so called "test-only" words and "unsolvable" words. "Test-only" are the polysemous words that are present only in the test data. Such words can be recognized by a classifier but cannot be correctly disambiguated by any classification model constructed only from the given set of training examples. That is why, no test examples are constructed for such words. However, statistics on monosemous and "test-only" words are used for calculating the first sense baseline accuracy (FSB) and the first sense dictionary-based baseline accuracy (WNFS) [4] over the available data sets, assuming that examples of such words are solved by a back-off strategy for the corresponding classifier. The "unsolvable" words are polysemous words whose correct sense annotation tags are present only in the test data. It is clear that such words can never be disambiguated correctly by any classifier learned from the given training data. Since "unsolvable" words cannot be recognized by a classifier, we generate examples of these words and the system uses such data for calculating a so called "Best Case" baseline that determines the upper bound of the classification accuracy that can be theoretically achieved on this test set by a classifier trained on the given training set.

Finally, for each word in the dictionary the list of word senses is rearranged—from the most frequent to the least frequent one. The user can determine a manner in which to carry out such a rearrangement based on the sense frequency data available in the dictionary, the sense frequency data calculated over the training set or by mixing both types of such data. In the last case the sense frequency data calculated over the training data is applied only for the rearrangement of senses which have no associated frequencies data in the dictionary. This rearrangement is then used for defining classes of training and testing examples (see next subsection for details).

3.2. Data Generator Module

The Data Generator module is responsible for extracting various features and constructing from them training and test examples for solving the all-words WSD task based on a set of parameters specified by the user. Examples are created for each occurrence of each polysemous word found in the corresponding text data (i.e., nouns, verbs, adjectives and adverbs), which are not "test-only" words. We assume that examples with monosemous and "test-only" words will be classified by selecting the first sense of the word from the corresponding WordNet-based dictionary (which is, for example, the backoff strategy used by the IMS system [10]).

3.2.1. Representation of Examples

An example is represented by a target word (and its POS category), by words (and their POS categories) in specific positions near the target word (so called "context words"), as well as the target word class (i.e., its correct sense):

$$w_{\text{target}}, POS_{\text{target}}, [w_{i-l}, POS_{i-l}, \ldots, w_{i-1}, POS_{i-1}, w_{i+1}, POS_{i+1} \ldots, w_{i+r}, POS_{i+r}], class(w_{\text{target}})$$

where i is the position of the target word in a sentence, l and r are, respectively, the left and the right boundaries of the context window and $class(w_{\text{target}})$ codes the correct meaning of the target word. The user has several options for constructing the target word context—besides a possibility to define the window boundaries, she is also able to specify specific POS categories of the context words to be included into example representation as well as whether the context window can include words from adjacent sentences or not.

3.2.2. Representation of Target and Context Words

The system provides several options for representing words (target and context) as well as for the target word class—as plain text, numeric identifier or as embedding vectors. The POS categories can be also represented by plain text or as binary vectors. The plain text representation of all elements allows presenting examples in an easily readable form that is assumed to be used for inspection purposes. The use of sense embeddings for representing the target word meaning allows for reformulating the all-words WSD classification task as a regression problem.

The WSD Studio possesses two unique features that distinguish it from all similar WSD systems—the ability to construct a special (so called "compressed") representation for word embeddings and the ability to construct a special representation for target word senses. The compressed representation of word embeddings replaces the distributed representation of a word (vector of real numbers) by only one real number, which significantly reduces the dimensionality of training and test data sets. Two types of embedding compression are supported—the first is implemented by a calculation of the cosine similarity between each embedding vector and the unitary vector (i.e., a vector whose arguments are all equal to 1) of the same dimensionality. The second approach creates the compressed representation of *all context words* by calculating the cosine similarity between their embedding vectors and the embedding vector of the target word (see [9] for a more detailed description).

3.2.3. Representation of the Target Word Class

The coding of the target word class used in the WSD Studio is based on our new formulation of the all-words WSD task. In all other WSD supervised systems each sense of a target word is considered as a unique class, which has its own encoding specific for that word. In our approach we propose a unified interpretation of a word sense class, which does not depend on the concrete word—the class of a target word is interpreted as the place of the corresponding word sense in an ordered sequence of the relative sense frequencies associated with the word in the modified WordNet-based dictionary. In other words, the most frequently used sense of each target word is marked as *class 1*, the next more frequent word sense is marked as *class 2* and so on until the least frequent one is reached. The overall number of classes is defined by the maximum number of possible senses for a word in the dictionary (which is equal to 75 for our WordNet-based dictionary). The proposed approach allows the user to implicitly define the number of classifiers that she/he wants to construct from the available data for solving the all-words WSD task (see next subsection for more detailed explanations).

It should be noted that, according to this interpretation, the disambiguation of an unknown open class word is a two-step process—at the first step the class associated with the sense of the test word is determined by a classifier and then the proper sense is retrieved from the dictionary via a WordNet sense identifier that corresponds to the class for the test word.

3.3. Data Writer Module

The DataWriter module is responsible for saving the constructed examples as a set of training and test data files intended to be used by different classifiers. The number of files for storing training and test examples depends on the saving strategy chosen by the user. The default strategy is to store all examples in two files (one—for training and one—for test examples). This approach allows for the creation of a single classifier that is able to disambiguate all open class words occurring in the text. An alternative possibility is to group examples based on a concrete target word or even

on a POS category of the target word. In such a way the user can create a set of word-specific training and test files that can be used for creating traditional word expert classifiers. An ability to group examples according to POS categories of target words is situated between these two extremes. The choice of this saving strategy leads to creating four sets of examples that can be used further for constructing four different classifiers specializing in solving the WSD task for nouns, verbs, adverbs and adjectives. Such a flexible approach allows for the generation of different classification models for different POS categories, depending on the quality and quantity of available examples for each category. An finally, it should be mentioned, that all such created data sets can be saved either as .txt or .arff files, which makes it possible to use them directly in such machine learning environments as WEKA (https://www.cs.waikato.ac.nz/ml/weka/) or Orange (https://orange.biolab.si/).

4. Experiments

The main objective of the conducted experiments was to evaluate the potential of the example representation created by means of the WSD Studio for solving the all-word WSD task by different types of classifiers. At the beginning we experimented with two types of classifiers—neural network based and ensemble based ones. The fully connected neural networks (FCNN) were built with TensorFlow (https://www.tensorflow.org/)—an open source machine learning framework. As the ensemble based classifiers we have selected Random Forest in its scikit-learn implementation (http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html) and the Deep Forest model [22] in its GCForest implementation (https://github.com/kingfengji/gcForestDeep).

4.1. Comparison with a Knowledge-Based WSD System

The first set of experiments was aimed to compare the behaviour of such classifiers with a knowledge-based WSD system developed by our colleagues [23]. We conducted experiments on the same training and test data extracted from SemCor 3.0 (http://web.eecs.umich.edu/~{}mihalcea/downloads/semcor/semcor3.0.tar.gz). We also used the same WordNet dictionary wnet30_v203.lex (https://github.com/asoroa/ukb/blob/master/src/README) and the same word embeddings with 300 dimensions (WN30WNGWN30glConOneGraphRelSCOne-synsetEmbeddings.bin downloaded from http://bultreebank.org/en/DemoSem/Embeddings). Some basic statistics of this text data at the level of words are shown in Table 1.

Table 1. Some basic statistics on the text data sets used in the experiments.

Data Set	Number of Word Occurrences	Number of Occurrences of Polysemous Words	Number of Occurrences of Monosemous Words	Maximum Number of Senses Per Word	Number of Occurrences of "Test-Only" Words	Number of Occurrences of "Unsolvable" Words
Training text data	134,372	115,095	19,277	31	-	-
Test text data	49,541	42,296	7245	23	2042	3594

As we have already discussed earlier, the WSD Studio constructs examples only for polysemous words, which are not of the "test-only" type—each occurrence of such words in the text data is converted to an example.

The detailed description of all experiments with the mentioned above data sets are presented in Reference [9]. Table 2 presents only the summary of the best results achieved during the experiments. The training examples were split to four sets according to the POS category of the target word (i.e., nouns, verbs, adjectives and adverbs) and each set was used for learning a classification model by means of the corresponding learning algorithm. The description of an example included an embedding of the target word as well as embeddings and POS categories of the context words. The size of the context window was set to 10 words—5 words left and 5 words right to the position of a target word in the sentence. The cosine similarity between the target and context words was used as the method for word embedding compression. The reported accuracy was calculated on all examples—test

set examples of monosemous and "test-only" polysemous words were classified by means of the WNFS heuristic.

As can be seen, the Random Forest classifier trained on the examples containing the uncompressed variant of word embeddings achieved the best accuracy, slightly better than the WordNet First Sense Baseline (WNFS) (It should be noted, that the calculation of WNFS statistics used the information both on the frequency of a word sense and on the word's POS category) and significantly higher than the accuracy of the knowledge-based WSD system tested on the same data. It is worth noting the very competitive results achieved on the compressed representation of word embeddings, having in mind that the dimensionality of such compressed representation of examples is about 300 times more compact than the uncompressed one.

Table 2. The best classification accuracy (in %) achieved on all examples.

Classifier	Random Forest(100) + "Full" Word Embeddings + WordNet-Based Class Ordering	Random Forest(100) + "Compressed" Word Embeddings + WordNet-Based Class Ordering	Knowledge-Based Classifier [23]	WNFS
Accuracy (%)	**75.85**	75.67	68.77	75.71

4.2. What Words Should be Included into the Context Window?

In the previous experiments, words included into the context window were restricted only to open class words. The next set of experiments was aimed at evaluating the influence of the inclusion of functional words (i.e., conjunctions, prepositions, etc.) into example description. In these experiments we used the same separation of text data into training and test files, however the sentences now included the functional words as well. The GloVe embeddings [24] were used for encoding all words. In all experiments with neural networks we applied the same set of parameters—Sofmax evaluation function, ReLU activation function, learning rate set to 0.003, dropout set to 0.5 and Adam as an optimizer. The FCNN classifiers used in all experiments had the following architectures: 3400 input nodes, 1000 nodes at the first hidden layer, 1000 nodes at the second hidden layer and 75 nodes at the output layer.

Table 3. The best classification accuracy (in %) achieved on uncompressed examples.

POS of the Target Word	GCForest		FCNN		WNFS Baseline
	Context: Open Class Words Only	Context: All Words	Context: Open Class Words Only	Context: All Words	
ADJ	81.35	81,46	81,20	81.24	81.20
NOUN	73.06	73.45	73.38	73.35	73.38
VERB	57.20	59.31	56.66	56.93	56.66
ADV	77.47	79.39	75.84	76.71	75.84
Overall	70.59	**71.54**	70.49	***70.62***	70.49

The results of these experiments are presented in Table 3. The accuracy values shown in the table were calculated only on the test examples created by the system (i.e., excluding monosemous and "test-only" words).

It can be seen that the inclusion of the functional words into the context window leads to increasing the classification accuracy of both classifiers, which now both beat the WNFS baseline. That is why in the next experiments we used a representation of examples including both open class and functional words.

4.3. Evaluation of the Influence of Word Embedding Compression

The dimensionality of data with the uncompressed representation of embeddings used in the previous experiments was 4400. The compression of word embeddings allows to reduce the dimensionality of the data only to 111 attributes. Table 4 shows how such a drastic reduction influences the classifiers'

accuracy. The FCNN architecture used for classifying the compressed test data comprised the following layers: 111 (input)—1000—1000—75 (output).

Table 4. The best classification accuracy (in %) achieved on compressed examples.

POS of the Target Word	GCForest		FCNN		WNFS Baseline
	Compressed	Uncompressed	Compressed	Uncompressed	
ADJ	81.20	81.46	81.20	81.24	81.20
NOUN	73.38	73.45	73.38	73.35	73.38
VERB	57.08	59.31	57.26	56.93	56.66
ADV	76.20	79.39	75.84	76.71	75.84
Overall	*70.63*	**71.54**	**70.65**	70.62	70.49

It can be seen that the proposed method for word embedding compression works surprisingly well—both classifiers trained on such representations of examples have achieved classification accuracy higher than WNFS baseline, which is very difficult to beat. Moreover, while the accuracy of GSForest slightly decreased (less than 1%) due to the embedding compression, the accuracy of FCNN even increased slightly. These results have proved that the proposed method for word embedding compression preserves enough information, allowing a classifier to disambiguate effectively unseen open class words.

The analysis of the confusion matrices of the classifiers used in all experiments has shown that they all have a tendency to classify most examples into the class that occurs most frequently in the training set. This MFS bias is a well-known fact for all WSD systems (see, i.e., [25]) and is caused by a highly skewed distribution of classes in the training data. So, it is reasonable to expect that better results could be achieved by a classifier that is able to pay more attention to examples from less frequent classes. In order to test such a hypothesis, we have selected the JRip classifier—a WEKA implementation of the RIPPER algorithm—for rule induction, including heuristic global optimization of rule sets [26]. The algorithm incrementally learns rules for less presented classes and classifies an example to the most frequent class by means of a default rule. Because of the restriction of the available memory, we applied JRip only to the compressed version of the data used in the previous experiments. Table 5 presents the accuracy of the JRip classifier in comparison with that of GCForest and FCNN.

Table 5. The classification accuracy (in %) achieved on the compressed examples.

	GCforest	FCNN	JRIP	Examples	WNFS Baseline
ADJ	81.20	81.20	81.20	7196	81.20
NOUN	73.38	73.38	73.38	19450	73.38
VERB	57.08	57.26	**62.53**	10781	56.66
ADV	76.20	75.84	**77.68**	2989	75.84
Overall	70.63	**70.65%**	**72.20**	40416	70.49

The results presented in Table 5 have fully confirmed our expectations about the potential of the JRip algorithm as an effective classifier for solving the all-words WSD task. It significantly outperforms both the neural net based and ensemble based classifiers on two of four data sets with the compressed representation of examples, as well as the WNFS baseline. Moreover, it achieved the highest average accuracy even in comparison with the best result of the GCforest classifier achieved on the uncompressed data sets. The results of the experiments presented in this subsection allow us to conclude that the proposed approach for word embeddings compression is a very promising way for significantly reducing the dimensionality of data used for solving the all-words WSD task,

with inducing a practically insignificant decrease in classification accuracy of classifiers learned from such compressed data. The approach also allows for the significant extension of the scope of possible classifiers to be applied to this task.

4.4. Evaluation of the Influence of the Data Set Granularity

As it has been already mentioned, a unique feature of the WSD Studio is the ability to construct training and test data sets with different granularity—the coarsest set contains examples of all open class words. The training set of such granularity can be used for generating a single classification model that is able to disambiguate all open class words. The middle level of the data set granularity are data sets joining examples (occurrences of open class words) belonging to the same POS category—nouns, verbs, adjectives and adverbs. The classifiers learned from such data sets can be considered as experts for disambiguating all words belonging to the corresponding POS category (that is why we call them "POS experts"). The finest granularity can be realized by grouping examples according to the combination of the target word and its POS category. Each data set of such granularity contains all examples of a given open class word belonging to a concrete POS category. The existing supervised WSD systems are able to create only data sets of this granularity and the classifiers learned from such sets are called "word experts." In the experiments presented in this subsection we tried to evaluate the dependence between the overall accuracy of classifiers and the level of granularity of training subsets used for training these classifiers.

Table 6. The classification accuracy (in %) achieved by word and POS experts.

	Word Experts (256 FCNN Classifiers)		**POS Experts (4 FCNN Classifiers)**		**WNFS Baseline**
	Uncompressed	**Compressed**	**Uncompressed**	**Compressed**	
ADJ	81.80	81.61	81.24	81.20	81.20
ADV	78.82	78.45	76.71	75.84	75.84
NOUN	73.56	73.23	73.35	73.38	73.38
VERB	59.18	58.66	56.93	57.26	56.66
Overall	**71.58**	***71.22***	70.62	70.65	70.49

The experiments were conducted on the same training and test data by means of FCNN classifiers having the same architecture as described above. The number of all different pairs: polysemous word—word POS category in the training data was 5840; among them 320 were pairs for adverbs, 1255—for adjectives, 1446—for verbs and 2819—for nouns. We decided to construct classifiers (word experts) only for pairs that had at least 60 occurrences in the training data (i.e., were represented by at least 60 training examples). This reduced the number of different pairs to only 256: 34—for adverbs, 42—for adjectives, 65—for verbs and 115—for nouns. No classification models were constructed for the rest 5584 pairs of the polysemous words and their POS categories. Each training example containing such a pair was classified by means of the WNFS heuristic.

The experiments were conducted both with uncompressed and compressed data and their results are presented in Table 6. For purposes of the comparison the table also shows the accuracy of four "POS experts" with the same architecture and parameters.

As it could be expected, the increase of data granularity leads to the increase of the overall accuracy of classifiers learned from the data. Such an increase is observed both for uncompressed and compressed data even though the amount of such an increase is not significant (about 1%).

4.5. Comparison with the State-of-the-Art Supervised WSD Systems

The last set of experiments reported in this paper was to compare the behaviour of classifiers trained on the training examples generated by the WSD Studio with some state-of-the-art WSD systems

trained on the same text data sets. For such a comparison we used the SemCor [3] corpus version 3.0 (web.eecs.umich.edu/~mihalcea/downloads.html) as the training text data and the concatenation of the Senseval and SemEval data sets [16] as the test text data. The GloVe word embedding model [24] was used for the representation of data. The overall number of occurrences of different open class words in this test data is 7253, out of which 4300 are nouns, 1652 are verbs, 955 are adjectives and 346 are adverbs. The analysis conducted by the WSD Studio has also shown that 1045 of them are the occurrences of monosemous words and 222—of "test-only" words. So these 1267 examples were classified in our experiments by applying the WSNF heuristic. The remaining occurrences of the polysemous words (5986) were used by the WSD Studio for generating test examples.

In the experiments we used only the compressed representation of examples. The training and test sets of examples were generated in two variants with different data granularity—in the first case such sets were created at the level of POS experts (i.e., four different subsets for nouns, verbs, adjectives and adverbs) and in the second case—at the level of word experts (i.e., for each pair: word—POS category of the word). The POS experts were created by means of the JRip algorithm and the word experts—by FCNN classifiers with the same set of parameters as used in all previous experiments.

The SemCor corpus contains 2122 combinations of the polysemous words and their POS categories, however only 446 of them are represented by more than 60 examples. We generated word expert classifiers only for those 446 pairs and for the classification of testing examples belonging to the rest (1676) pairs we applied the WNFS heuristics as the back-off classification strategy. The results of the experiments are presented in Table 7.

Table 7. Comparison of the WSD Studio with the state-of-the-art supervised WSD systems (adapted from [20]).

Training Corpus	Test Corpus	Systems	Accuracy (%)
SemCor	Concatenation of Senseval and SemEval collections	IMS	68.4
		IMS + emb	69.1
		IMS-s + emb	**69.6**
		Context2Vec	69.0
		MFS	64.8
		WNFS	65.2
		WSD Studio + JRip (POS experts)	*65.3*
		WSD Studio + FCNN (Word experts)	*66.8*

5. Discussion

The results presented in Table 7 show that the accuracy achieved by both classifiers trained on the examples generated by the WSD Studio is higher than the WNFS baseline but still lower than the accuracy of the best supervised WSD systems. However, it should be noted that based on the results of previous experiments we could expect that the accuracy of JRip-based word experts trained on the same compressed representation may become higher than that of FNCC-based experts. We also expect an increase in accuracy when the uncompressed representation of examples will be used instead of the compressed one.

A possible direction for raising the classification accuracy of classifiers trained in the examples generated by the WSD Studio is also to use more "advanced" word embeddings as, for example, it was proposed in Reference [16]. We will check both hypotheses in the nearest future.

The surprisingly good results achieved by classifiers of different types trained on the compressed representation of examples constructed by the WSD Studio are another topic worth discussion. Our current hypothesis is that such a compressed representation preserves most of the syntactic and semantic information in the word embeddings by measuring similarity between target and context words.

Another interesting question related to our approach is the dependence of classification accuracy on the level of granularity of data sets generated by the WSD Studio. Currently we think that the decrease of granularity leads to amplifying the existing imbalance between classes on the one hand and to an increase of the number of classes in the data set on the other. As a result, the boundaries between classes become more complex and, hence, more difficult to be learned correctly by classifiers. Deeper answers on both questions are expected to be found after an error analysis, which is our ongoing work.

The last problem that we would like to discuss in this paper is possible ways to enrich the current representation of examples used in our system. A simple comparison between the bag-of-words approach for example representation used in all IMS-based WSD systems with the collocation approach applied by us has shown that we do not use explicitly information about collocations between context words. It is very probable that the presence of such information contributes to higher results achieved by such systems. That is why our ongoing research is oriented to developing natural and effective ways for integrating the collocation information into the existing representation of examples used by the WSD Studio, as well as to invent collocation compression methods compatible with the one applied to the compression of word embeddings.

6. Conclusions

In this paper we have presented the WSD Studio—a flexible system for extracting features and creating the representation of examples used for solving the all-words WSD task. The system provides support for multiple data formats and has a highly configurable data generation process. The WSD Studio is characterized by two unique features distinguishing it from all similar WSD systems—the ability to construct a special compressed representation for word embeddings and the ability to construct training and test sets of examples with different data granularity. The first feature allows generation of data sets with quite small dimensionality, which can be used for training highly accurate classifiers of different types. The second feature allows for the generation sets of examples that can be used for training classifiers specialized in disambiguating a concrete word, words belonging to the same POS category or all open class words.

One drawback of the system is that it currently operates only on Windows operating systems. Another drawback is that the size of the input data that can be processed by the system is limited by the machine's RAM since in the current implementation some operations require all data to be loaded into system memory.

Our future plans include extending the system architecture by a module for external plugins allowing development of third party modules that implement different input data formats, additional data generation algorithms and a larger set of supported output data elements.

WSD Studio is implemented in the .NET Framework 4.5, uses C# as a programming language and has a Windows forms-based user interface.

Author Contributions: The author contributions are as follows: conceptualization and methodology, G.A.; software, D.P. and S.K.; experimental investigation, D.P., S.K. and G.A.; formal analysis—G.A.; writing—original draft preparation, G.A. and D.P.; writing—review and editing, G.A.

Acknowledgments: Research described in this article was partially supported by the National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)," financed by the Ministry of Education and Science.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Navigli, R. Word sense disambiguation: A survey. *ACM Comput. Surv.* **2009**, *41*, 2. [CrossRef]
2. Fellbaum, C. WordNet and wordnets. In *Encyclopedia of Language and Linguistics*, 2nd ed.; Elsevier: Oxford, UK, 2005; pp. 665–670.

3. Miller, G.A.; Leacock, C.; Tengi, R.; Bunker, R.T. A semantic concordance. In Proceedings of the ARPA Workshop on Human Language Technology, Princeton, NJ, USA, 21–24 March 1993; pp. 303–308.
4. Pilehvar, M.T.; Navigli, R. A large-scale pseudoword-based evaluation framework for state-of-the-art Word Sense Disambiguation. *Comput. Linguist.* **2014**, *40*, 837–881. [CrossRef]
5. Pasini, T.; Navigli, R. Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 77–88.
6. Lesk, M. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th SIGDOC*; ACM: New York, NY, USA, 1986; pp. 24–26.
7. Camacho-Collados, J.; Pilehvar, M.H.; Navigli, R. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artif. Intell.* **2016**, *240*, 36–64. [CrossRef]
8. Agirre, E.; Soroa, A. Personalizing Pagerank for Word Sense Disambiguation. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, Athens, Greece, 30 March–3 April 2009; pp. 33–41.
9. Agre, G.; Petrov, D.; Keskinova, S. A new approach to the supervised word sense disambiguation. In *Lecture Notes in Computer Science*; Springer: Berlin, Germany, 2018; Volume 11089, pp. 3–15.
10. Zhong, Z.; Ng, H.T. It Makes Sense: A wide-coverage Word Sense Disambiguation system for free text. In Proceedings of the ACL System Demonstrations, Uppsala, Sweden, 13 July 2010; pp. 78–83.
11. Cortes, C.; Vapnik, V.N. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
12. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
13. Taghipour, K.; Ng, H.T. Semisupervised word sense disambiguation using word embed-dings in general and specific domains. In Proceedings of the NAACL HLT, Denver, CO, USA, 31 May–5 June 2015; pp. 314–323.
14. Rothe, S.; Schutze, H. Autoextend: Extending word embeddings to embeddings for syn-sets and lexemes. In Proceedings of the ACL 2015, Beijing, China, 26 July 2015; pp. 1793–1803.
15. Iacobacci, I.; Pilehvar, M.H.; Navigli, R. Embeddings for word sense disambiguation: An evaluation study. In Proceedings of the ACL, Berlin, Germany, 7–12 August 2016; pp. 897–907.
16. Koprinkova-Hristova, P.; Popov, A.; Simov, K.; Osenova, P. Echo state network for word sense disambiguation. In *Lecture Notes in Computer Science*; Springer: Berlin, Germany, 2018; Volume 11089, pp. 73–82.
17. Popov, A. Neural Network Models forWord Sense Disambiguation: An Overview. *Cybern. Inf. Technol.* **2018**, *18*, 139–151.
18. Melamud, O.; Goldberger, J.; Dagan, I. Learning Generic Context Embedding with Bidirectional LSTM. In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL), Berlin, Germany, 7–12 August 2016; pp. 51–61.
19. Kageback, M.; Salomonsson, H. Word sense disambiguation using a bidirectional lstm. *arXiv* **2016**, arXiv:1606.03568.
20. Yuan, D.; Richardson, J.; Doherty, R.; Evans, C.; Altendorf, E. Semi-supervised word sense disambiguation with neural models. In Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016), Osaka, Japan, 11–16 December 2016; pp. 1374–1385.
21. Raganato, A.; Camacho-Collados, J.; Navigli, R. Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison. In Proceedings of the EACL 2017, Valencia, Spain, 3–7 April 2017; pp. 99–110.
22. Zhou, Z.-H.; Feng, J. Deep Forest: Towards an Alternative to Deep Neural Networks. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17), Melbourne, Australia, 19–25 August 2017; pp. 3553–3559.
23. Simov, K.; Osenova, P.; Popov, A. Using context information for knowledge-based word sense disambiguation. In *Lecture Notes in Artificial Intelligence*; Springer: Berlin, Germany, 2016; Volume 9883, pp. 130–139.
24. Pennington, J.; Socher, R.; Manning, C. Glove: Global vectors for word representation. In Proceedings of the Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; Volume 14, pp. 1532–1543.

25. Postma, M.; Izquierdo, R.; Agirre, E.; Rigau, G.; Vossen, P. Addressing the MFS Bias in WSD systems. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slovenia, 23–26 May 2016; pp. 1695–1700.
26. Cohen, J. Fast effective rule induction. In Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, CA, USA; 9–12 July 1995; pp. 115–123.